```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
FractionToCF <- function(numerator, denominator){
  num <- numerator; den <- denominator
  #
  i <- 0
  cf <- vector()
  if(num == 1) cf <- c(0, den)
  while(num > 1){
    i <- i+1
    jj <- floor(num/den)
    cf[i] <- jj
    temp <- num
    num <- den
    den <- temp-floor(temp/den)*den}
  return(cf)
}
#
# # # # # # # # # # # # # # # # # # # # # # #
#
# system = 1 Calkin-Wilf
# system = 2 driB
# system = 3 Yu-Ting
# system = 4 Yurramendi-1
#
CFToBinary.1.1 <- function(cf){
  binary.vector <- vector()
  for(k in 1:length(cf))
    binary.vector <- c(binary.vector, rep(k%%2, rev(cf)[k]))
  if(cf[1] == 0 & binary.vector[sum(cf)] != 0 |
     cf[1] != 0 & binary.vector[sum(cf)] == 0 )
    binary.vector[-1] <- 1 - binary.vector[-1]
  return(binary.vector)
}
#
CFToBinary.1.2 <- function(cf){
  binary.vector <- vector()
  for(k in 1:length(cf))
    binary.vector <- c(binary.vector, rep(k%%2, rev(cf)[k]))
  filter   <- rep(c(1,0), 15)[1:sum(cf)]
  binary.vector <- as.numeric(binary.vector == filter)
  if(cf[1] == 0 & binary.vector[sum(cf)] != 0 |
     cf[1] != 0 & binary.vector[sum(cf)] == 0 )
    binary.vector[-1] <- 1 - binary.vector[-1]
  #
  return(binary.vector)
}
#
CFToBinary.1.3 <- function(cf){
  binary.vector <- rep(0, sum(cf))
  binary.vector[c(1, cumsum(rev(cf)))] <- 1
  if(cf[1] == 0) binary.vector[sum(cf)] <- 0
  return(binary.vector)
}
#
CFToBinary.1.4 <- function(cf){
  binary.vector <- rep(1, sum(cf))
  binary.vector[cumsum(rev(cf[-1]))] <- 0
  return(binary.vector)
}
#
# # # # # # # # # # # # # # # # # # # # # # #
#
BinaryToLocation <- function(binary.vector){
  location <- strtoi(paste(binary.vector, collapse = ""), base = 2)
  return(location)
}
#
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
FractionToLocation.CF.1 <- function(system, numerator, denominator){
#
#   system = 1  Calkin-Wilf
#   system = 2  driB
#   system = 3  Yu-Ting
#   system = 4  Yurramendi-1
#
# Fraction --> Continued Fraction --> Binary --> Location
#
# Fraction --> Continued Fraction
#
  cf <- FractionToCF(numerator, denominator)
#
#              Continued Fraction --> Binary
#
  if(system == 1)  binary.vector <- CFToBinary.1.1(cf)
  if(system == 2)  binary.vector <- CFToBinary.1.2(cf)
  if(system == 3)  binary.vector <- CFToBinary.1.3(cf)
  if(system == 4)  binary.vector <- CFToBinary.1.4(cf)
#
#                                    Binary --> Location
#
  location <- BinaryToLocation(binary.vector)
#
  return(location)
}
#
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#
# Examples: Compute location of 9/5 in all four systems
#
FractionToLocation.CF.1(1, 9, 5) # 61
FractionToLocation.CF.1(2, 9, 5) # 55
FractionToLocation.CF.1(3, 9, 5) # 39
FractionToLocation.CF.1(4, 9, 5) # 57
#
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
```